Pengontrol Lampu via Missed Call Berbasis Mikrokontroler AVR AT90S2313

A Lamp Controller via Missed Calls Based On The Microcontroller AVR AT90S2313

Jazi Eko Istiyanto & Andik Dwi Kuntoro
Jurusan Fisika FMIPA UGM

ABSTRACT

Missed calls have a potential for low-cost control and monitoring of a remote instrumentation system as compared to SMS (Short-Message Service) and DTMF (Dual-Tone Multiple Frequencies). The design of an instrumentation system model in which a set of lamps can, remotely, be turned on/off just by issuing a set of unanswered calls and the programming of the AT90S2313 microcontroller to serve the purpose of controlling and status monitoring of a set of six lamps (AC bulbs) are presented in this paper. Two cellular phones are used: one as the calling phone, and the other as the receiving end (the transceiver). The transceiver is attached to the microcontroller circuitry. The calling phone issues two sets of RINGs and the transceiver is programmed to disregard any RING. The first set of RINGs is interpreted as a command to select one of the six lamps, then the second set of RINGs constitutes a command to turn the selected lamp on/off. This is possible because the cellular system records the number of RINGs before a call qualifies as a missed call.The system has successfully been built, tested, and proven to work well. The limitation is, apart from the GSM signal availability, the inability of the current system to verify/authenticate the calling phone number.

Keywords: Missed call, microcontroller, *AVRAT90S2313*

PENDAHULUAN

Istiyanto & Effendy (2004) telah membuat sebuah sistem untuk menghidupkan/mematikan, dari jarak jauh, sejumlah LEDs (*Light-Emitting Diodes*) yang terhubung dengan suatu rangkaian berbasis mikrokontroler AT89C51. Untuk menghidupkan/mematikan sejumlah LED, pengguna mengirimkan sebuah perintah dalam bentuk SMS (*Short-Message Service*) ke telepon seluler yang terhubung pada rangkaian mikrokontroler tersebut.

Sistem buatan Istiyanto & Effendy (2004) tersebut telah dikembangkan oleh Istiyanto & Purwadi (2005) agar dapat, dari jarak jauh, memonitor hasil pengukuran temperatur. LED telah digantikan dengan sensor temperatur LM555, tentu saja dengan modifikasi rangkaian seperlunya. Perintah yang dituliskan dalam SMS juga telah disistematisasikan dan interpretasinya telah didukung oleh suatu algoritma *parsing* dan *tokenising* yang dikenal dalam teknik kompilasi. Pengguna juga dapat menspesifikasikan kapan dan setiap berapa interval waktu laporan hasil pengukuran harus dikirimkan. Pekerjaan Istiyanto & Purwadi (2005) kemudian dimodifikasi oleh Istiyanto & Alrosyid (2005) untuk memonitor tidak hanya temperatur tetapi juga kejadian-kejadian yang dapat memicu alarm. Alarm ini diaktifkan sebagai akibat dari hasil pembacaan temperatur yang melampaui nilai tertentu. Sistem ini juga masih menggunakan SMS sebagai sarana komunikasi jarak jauh.

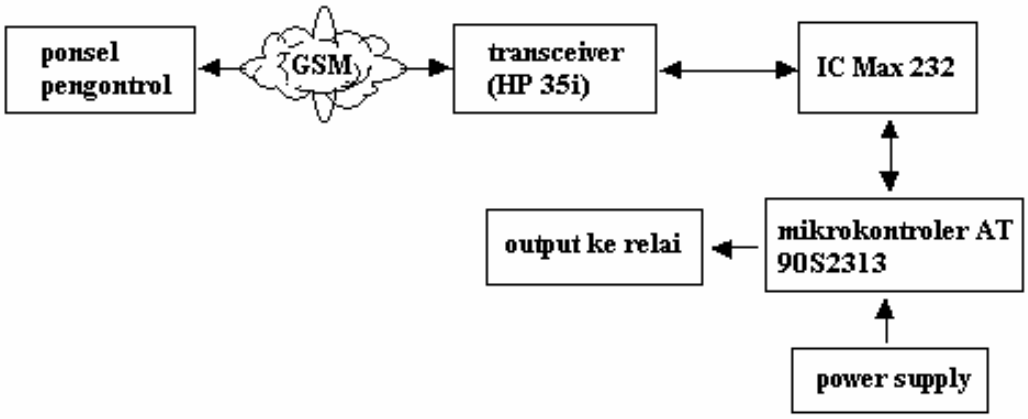
Istiyanto & Hakim (2007) telah mengimplementasikan suatu prototipe sistem otomasi rumah berbasis DTMF. Prototipe ini memungkinkan pengguna menghidupkan/mematikan lampu AC dan mengunci/membuka kunci sebuah pintu dengan mengirimkan perintah yang disandikan sebagai DTMF. Tergantung pada regulasi pemerintah, DTMF dapat lebih murah/lebih mahal dari pada SMS. Suatu alternatif untuk SMS dan DTMF adalah *missed call*, yang gratis, tetapi hanya dibatasi maksimum 6(enam) kali RING sebelum diakhiri oleh operator. Istiyanto & Wibhowo (2007) telah membuat suatu prototipe pengendali motor DC berbasis *missed call*. *Missed call* didefinisikan sebagai panggilan tak terjawab. *Missed call*, seperti halnya DTMF, juga dikenal pada pesawat PSTN (*Public Switch Telephone Network* – telepon rumah) tetapi untuk kemudahan dan kepraktisan pengujian prototipe, telepon selular

saja yang telah digunakan. Perintah pengendalian, dalam hal ini, disandikan sebagai banyaknya RING sebelum suatu panggilan divonis sebagai *missed call*.

METODE

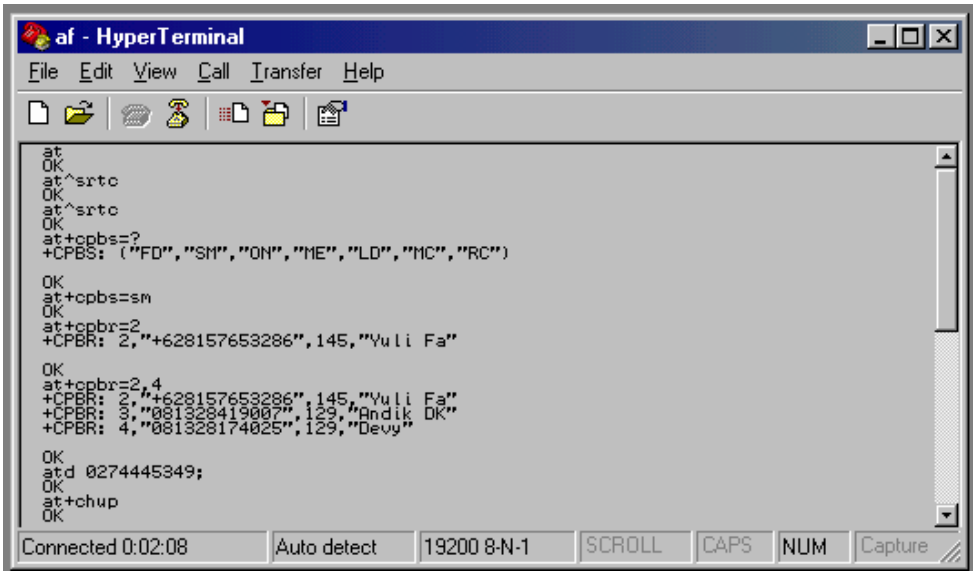
Diagram blok rancangan alat yang dilaporkan pada makalah ini dapat dilihat pada Gambar 1. Pada Gambar 1 dilukiskan bahwa ponsel pengontrol dihubungkan dengan ponsel transceiver melalui operator GSM, data missed call dari ponsel tranceiver diatur secara serial oleh IC Max 232 untuk dikirim ke mikrokontroler, dan data missed call diterjemahkan oleh mikrokontroler menjadi aksi untuk menghidupkan/mematikan 6 buah lampu AC (output ke relay), serta mikrokontroler mengirim perintah ke ponsel transceiver untuk memberikan status laporan ke ponsel pengontrol.

Gambar 2 melukiskan diagram skematik rangkaian elektronik yang tersusun atas mikrokontroler AT90S2313 (Atmel 2002), IC MAX232 (Maxim 2004), dan 6 (enam) buah relai. IC MAX 232 tersambung dengan port DB-9 dari koneksi RS-232. MAX232 dihubungkan dengan port PD (PD4 dengan T1IN, PD3 dengan R2OUT, PD0 dengan R1OUT, PD1 dengan T2IN) dari AT90S2313. Sedangkan relai disambungkan dengan port PB (PB5, PB4, PB3, PB2, PB1, dan PB0) dari AT90S2313. Makalah ini menekankan pada program bahasa rakitan yang ditanamkan di dalam AT90S2313 sebagai bagian dari rangkaian pada Gambar 2, agar dapat berperanan sesuai dengan rancangan Gambar 1.



Gambar 1. Diagram blok alat kontrol lampu berbasis ponsel.

Gambar 2. Diagram skematik sistem kontrol dengan *missed call*.



Gambar 3. Tampilan window hyperterminal.

Perintah AT (*AT Command Set*)

Perintah AT adalah deretan karakter yang diberikan dari Data Terminating Equipment (DTE) ke Data Communicating Equipment (DCE) (Siemens 2001). Ponsel GSM dapat dioperasikan menggunakan perintah AT via antarmuka serial (kabel data atau koneksi inframerah, ataupun Bluetooth). Perintah AT ini berbeda untuk spesifikasi GSM ponsel yang berbeda (Siemens 2001).

Perintah harus diawali dengan karakter "AT" dan diakhiri dengan "<CR>" (0X0D). Hasil dari perintah ditampilkan dengan "OK" atau "ERROR"(Siemens 2001). Perintah AT bisa digunakan untuk melakukan calling, mengirim sms, pengaturan nada dering panggilan, mengetahui nama pabrikan ponsel, versi GSM ponsel, dan masih banyak lagi. Salah satu software yang digunakan untuk menguji AT Command adalah Windows Hyperterminal (Gambar 3) yang tersedia bersama Windows Installer. Beberapa perintah AT, diantaranya adalah :

AT + CPBS = phonebook; phonebook dapat dispesifikasi dengan SM (SIM telephone book), LD (Last dialing number), ME (Mobile equipment telephone book). MC (Missed dialing number), RC (Received call list) atau DC (Dialled call list).
AT + CPBR = membaca phonebook.
ATD <m><n>; = melakukan dial ke lokasi memori phonebook <m>, nomor telepon <n>.
AT + CHUP = mengakhiri calling / panggilan
AT + SRTC = mengatur nada dering baik untuk mengaktifkan, menonaktifkan, ataupun memilih mode nada dering panggilan.

HASIL DAN PEMBAHASAN

Cara kerja alat

Pembahasan cara kerja alat ditekankan pada fungsi alat kontrol ini untuk menghidupkan/mematikan lampu AC (alat elektronik) dari jarak jauh saja. Alat kontrol jarak jauh ini berpusat pada mikrokontroler yang memiliki 7 buah pin D (PD0 – PD6) dan 8 buah pin B (PB0 – PB7) (Gambar 2). Pin B berfungsi sebagai output untuk menghidupkan/mematikan lampu. Sedangkan pin D terhubung langsung dengan kaki – kaki pada IC Max 232. PD0 terhubung dengan R1OUT (Gambar 2) yang berfungsi sebagai saluran penerima data UART, PD1 terhubung dengan T2IN berfungsi sebagai saluran pengirim data UART. Adapun PD3 terhubung dengan R2OUT yang berfungsi sebagai RE (*Read Enable*), sedangkan PD4 dengan T1IN yang berfungsi sebagai WE (*Write Enable*).

Alat kontrol bekerja melalui tahapan-tahapan : Ponsel pengontrol akan melakukan dua kali panggilan secara *missed call* ke ponsel *transceiver*. Satu panggilan *missed call* dapat berupa 1 sampai dengan 6 nada dering. Mikrokontroler siap menangani setiap panggilan masuk ke ponsel *transceiver*. Ketika ada panggilan masuk, mikrokontroler akan mengambil data *missed call* yang berupa 'ring'. Data diambil oleh mikrokontroler satu per-satu (per karakter) yaitu 'r', 'i', 'n', dan 'g' dengan format kode karakter 8 bit. Jumlah data *missed call* pertama diterjemahkan oleh

mikrokontroler untuk menentukan lampu yang akan dikontrol. Ring n kali diartikan sebagai perintah untuk memilih lampu ke n, n = 1, …, 6. Lampu yang dikontrol melalui *missed call* pertama, kemudian dihidupkan/dimatikan dengan *missed call* kedua. Ring satu kali diterjemahkan sebagai perintah menghidupkan lampu terpilih, dan ring dua kali diterjemahkan sebagai perintah mematikan lampu terpilih. Setelah tugas menghidupkan/mematikan selesai, mikrokontroler akan memberi perintah ke ponsel *transceiver* untuk melakukan panggilan balik ke ponsel pengontrol sebagai tanda bahwa pengontrolan telah selesai dilaksanakan.

Inisialisasi port I/O

Inisialisasi port berfungsi untuk menset fungsi port I/O. Bit pada register kontrol DDRB diisi dengan nilai '1' semua karena port B akan difungsikan sebagai output. Register kontrol UBRR digunakan untuk pengaturan agar *Baud Rate* sistem mikrokontroler sama dengan *Baud Rate* ponsel. Register UCR (UART *Control Register*) diset untuk mengaktifkan UART dalam pengiriman dan penerimaan data.

Memilih memori *phonebook*

Karena *phonebook* tersimpan di beberapa lokasi referensi, maka alamat memori harus dispesifikasi. Perintah AT untuk memilih *phonebook* adalah AT+CPBS (*Control Phone Book Selection*). Perintah AT+CPBS ini disimpan sebagai data di lokasi memori dengan *assembler directive* DB. Alamat memori program tersebut dimuat dalam register pointer ZL (R29).

Karena ZL merupakan *general purpose register* 8 bit, sedangkan alamat pada memori program 2 byte (16 bit), maka alamat asli pada memori program harus dikalikan dua. Dengan perintah LPM (*Load Program Memory*), yang berfungsi mengambil data pada program memori yang alamatnya dimuati di register ZL (R29), data langsung dipindahkan ke R0. Data pada alamat R0 kemudian dipindahkan ke register *txbyte* (R19) untuk dikirim ke ponsel melalui fitur UART dari mikrokontroler. Setelah satu karakter terkirim, isi dari register ZL (alamat pada program memori) ditambah satu untuk menunjuk byte berikutnya dalam memori program (Gambar 4). Setelah perintah AT terkirim, mikrokontroler akan menunggu respon dari ponsel berupa karakter 'OK' menandakan proses pemilihan memori penyimpan *phonebook* telah selesai (Gambar 5).

Memilih nomor telepon pada memori

Nomor telepon yang dipilih adalah nomor ponsel pengontrol yang disimpan di lokasi SRAM. Nomor ponsel ini akan dihubungi oleh ponsel *transceiver* sebagai tanda bahwa kontrol telah selesai.

```
daftar_phone: .db          "at+cpbs=sm",0x0d,0x0a, 0
init1: rcall    long_delay
rcall           pilih_memori
rcall           delay
rcall           cek_ok
brcs    init1
pilih_memori: ldi          zl,low(daftar_phone*2)   ;akses   alamat
        pada
ldi             zh,high(daftar_phone*2)  ;program memori
rjmp    send_command                     ;kirim data
send_command: lpm          ;akses  data  yang  beralamat  di  Z
        (R30),pindahkan ke r0
tst     r0    ;cek apakah data r0=0 (data terakhir)
breq    return      ;jika ya, kembali & jika belum
mov     txbyte,r0   ;pindahkan data ke register txbyte
rcall   tulis_tx    ;pindahkan ke register UART
adiw    zl,1        ; Zl=Zl+1
rjmp    send_command        ;lompat ke send_command
return: ret
```

Gambar 4. Prosedur untuk memilih memori phonebook.

```
cek_ok:  ldi      YL,RamStart
clr       YHramadr
clr       ZHramadr
again:   rcall    receive
cpi       rxbyte,'O'      ;cek apakah data diterima 'O'
brne      again           ;jika bukan, cek lagi
uart2ram:rcall    receive  ;terima respon/data lagi dari HP
cpi       rxbyte,'K'      ;cek apakah respon selanjutnya adalah 'K'
brne      cek_data_error          ;jika bukan, kembali kirim perintah AT

clc               ;jika ya, clear carry
          ret                     ;kembali
receive: sbis     USR,RXC         ;cek apakah data diterima komplet, jika
rjmp      receive         ;belum, cek lagi, jika sudah
in        rxbyte,UDR      ;pindahkan ke register rxbyte
          ret                     ;kembali
```

Gambar 5. Program untuk menerima data karakter ‘OK’.

```
init2:   rcall     long_delay
rcall     pilih_nomor             ;pilih nomor HP di phonebook
ldi       YLramadr,$61            ;byte pertam RAM
ldi       temp2,35        ;RAM yang digunakan sebanyak 35 byte
again2:  rcall    receive  ;panggil rutin terima data
cpi       rxbyte,'+'      ;apakah diterima '+'(data pertama dari +CPBR:
          1,"08122603521",129,"Jazi Eko")
brne      again2          ;jika bukan, terima data lagi
uart2ram2: rcall receive  ;panggil rutin terima data
 st       Y+,rxbyte       ;pindahkan data ke SRAM
 dec      temp2                   ;apakah data diterima semua (35 karakter)
             brne      uart2ram2          ;jika belum,terima data lagi
```

Gambar 6. Program pengiriman perintah AT+CPBR dan menyimpan respon dari ponsel ke lokasi SRAM.

Perintah AT+CPBR (*Control Phone Book Read*) memilih nomor telepon. Proses pengiriman perintah AT dan penerimaan respon dari ponsel pada dasarnya sama, yaitu dengan memanfaatkan fitur UART pada mikrokontroler. Perbedaannya adalah data respon yang didapatkan. AT+CPBS memberikan respon ‘OK’. AT+CPBR memberikan nomor ponsel yang dipilih dan nama pemilik. Perintah *at+cpbr=1* membaca nomor HP yang pertama kali tersimpan pada *phonebook*, yang memberikan, misalnya, *+CPBR: 1,"08122603521",129,"Jazi Eko"*. Respon ini kemudian disimpan di lokasi SRAM $61 (desimal 97) sampai $84 (desimal 132) (Gambar 6).

Pada Gambar 5 tampak bit RXC pada register USR mengindikasikan sebuah karakter telah diterima di UDR (register penyimpan data pada UART). Data tersebut dipindahkan Kemudian ke lokasi SRAM yang beralamat di register Y. Kemudian register Y ditambah dengan 1 untuk menyimpan data karakter berikutnya.

Pengecekan input

Mikrokontroler diprogram agar merespon *input* dengan teknik interupsi. Interupsi dipicu oleh adanya input dari ponsel yang berupa panggilan tak terjawab (*missed call*). Input *missed call* berupa deretan karakter ‘R’,’I’,’N’, dan ‘G’. Proses penerimaan data memanfaatkan fitur UART. Data dibandingkan

berturut-turut dengan karakter ‘R’,’I’,’N’, dan ‘G’. Jika masukan yang didapat bukan RING, maka program siap menerima interupsi berikutnya. Jika ada RING, maka isi register ringing (R23) akan ditambah dengan satu. Kemudian register I/O yang berhubungan dengan register *timer/counter* 16-bit akan diset (Gambar 7).

Timer/counter adalah pencacah biner yang jika bekerja pada frekuensi tetap disebut *timer* dan jika bekerja dengan frekuensi bervariasi disebut *counter*. Register I/O TCCR1B diatur sehingga nilai prescaler *timer/counter* 1 adalah CK/1024 yang berarti bahwa register cacahan yaitu TCNT1H dan TCNT1L akan terisi satu jika frekuensi kerja telah mencapai 1024 clock (1024 siklus). Kristal yang digunakan memiliki frekuensi kerja 8 MHz, sehingga satu clock memerlukan waktu 0.125 µdetik. Karena terdapat 16 bit register cacahan (TCNT1H dan TCNT1L), maka clock (sinyal detak) akan melebihi kapasitas pencacah (*overflow*) setelah $1024 \times 2^{16} \times 0.125$ µdetik = 8.38 detik. Saat clock (sinyal detak) yang diberikan sudah melebihi kapasitas pencacah (setelah 8.38 detik), maka pencacah akan memberikan sinyal *overflow* atau limpahan yang ditandai dengan bit ke-7 pada register TIFR bernilai ‘1’ (set).

```
cek_input: rcall cek_data_ring      ;cek apakah ada data ring masuk
  brcs      kembali                 ;jika bukan data ring, kembali
waiting_ring: ldi          temp,0b00000101          ;register          pencacah
          dihitung satu
    out     tccr1b,temp             ;jika mikro bekeja 1056 clock
loop1:  sbic     usr,rxc            ;apakah ada data masuk?
rcall     cek_data_ring ;jika ada apakah data itu data RING?
in        temp,tifr                      ;apakah data dicek selama 8 detik?
sbrs      temp,7                 ;jika belum selama 8 detik,
rjmp      loop1                  ;cek lagi apa ada data masuk
ldi       temp,(1<<7)
out       tifr,temp              ;jika sudah 8 detik,
inc       counter                ;hitung satu pada register counter
cpi       counter,3              ;apakah isi register counter=7
brne      waiting_ring   ;jika tidak, tunggu data ring selama 8s
return1: ldi      temp,(1<<7)             ;jika isi register counter=3,
out       tifr,temp              ldi       temp,0b00000110out
tccr1b,temp                      ldi       temp,$80

out       gimsk,temp             clr       temp

out       tcnt1h,temp
out       tcnt1l,temp
rjmp      tunggu_ring1   ;tunggu missed call kedua
cek_data_ring:  rcall     receive
                cpi       rxbyte,'R'        ;bandingkan data diterima dengan
          ‘R’
brne      cek_data_error          ;jika beda,kembali tunggu data lagi
                rcall     receive          cpi       rxbyte,'I'
                brne      cek_data_error   rcall     receive
                cpi       rxbyte,'N'                brne      cek_data_error
                rcall     receive          cpi       rxbyte,'G'
                brne      cek_data_error
                inc       ringing          ;ringing=ringing+1
clc
ret                               ;kembali
```

Gambar 7. Program untuk mengecek input data RING dari *missed call* yang pertama.

Pengesetan register TCCR1B dilakukan sebanyak 3 kali, sehingga program akan menunda selama 3 × 8,38 detik = 25,14 detik. Tunda ini berfungsi untuk menunggu dan menghitung 5 data RING sisa. Tahap selanjutnya, program akan menunggu *missed call* kedua. Jadi *missed call* kedua baru bisa dilakukan setelah 3 kali *overflow* atau setelah 25,14 detik setelah dial pada *missed call* pertama.

Pada tahap selanjutnya, program akan membandingkan isi register ringing (register penghitung jumlah RING) dengan angka – angka 1 s/d 6. Jika isi register ringing adalah n, misalnya, maka dengan menunggu jumlah RING pada missed call kedua akan menentukan lampu ke-n tersebut dihidupkan atau dimatikan. Jika jumlah RING pada missed call kedua adalah 1, maka lampu ke-n hidup dan jika jumlah RING – nya 2 maka lampu ke-n mati (Gambar 8). Pada Gambar 9, data yang diambil dari SRAM dimulai pada alamat *ramstart* ($60). Kemudian diambil data '+' yang terdapat pada alamat $61, 'C' pada $62, 'P' pada $63 dan seterusnya. Tentu saja data – data tersebut tidak dikirim ke register UART karena yang dibutuhkan hanya nomor HP saja (data antara karakter '"' pertama dan '"' kedua). Karena itu, isi register *txbyte* dibandingkan dengan karakter '"', jika belum sama, maka akan mengambil data pada alamat di atasnya. Jika telah sampai pada data '"', maka program akan mengakses program memori untuk mengambil perintah ATD dan dikirim ke HP melalui fitur UART. Tahap selanjutnya adalah pengaksesan SRAM yang beralamat di atas alamat yang mangandung data '"' tadi, yaitu data '0', '8', '1', dan seterusnya untuk dikirim ke ponsel *transceiver*. Setelah sampai karakter '"' kedua, pengiriman data respon perintah AT+CPBR ke ponsel *transceiver* dihentikan karena data yang diperlukan telah terpenuhi.

```
cpi         ringing,1
breq        ring1
cpi         ringing,2
breq        ring2
ring1:  rcall       cek_data_ring2      ;tunggu data ring pada MC kedua
rcall       waiting_ring1   ;tunggu ring lagi selama 8 detik
cpi         ringing1,1                  ;jika jumlah ring=1
breq        lampu1_hidup    ;lampu satu hidup
cpi         ringing1,2                  ;jika 2
breq        lampu1_mati                 ;lampu satu mati
ring2:  rcall       cek_data_ring2      ;tunggu data ring pada MC kedua
rcall       waiting_ring1   ;tunggu ring lagi selama 8 detik
cpi         ringing1,1                  ;jika jumlah ring=1
breq        lampu2_hidup    ;lampu dua hidup
cpi         ringing1,2                  ;jika 2
breq        lampu2_mati                 ;lampu dua mati
lampu1_hidup: cbi       portb,0             ;pb0 bernilai 0 (clear),lampu 1 nyala
rcall       report;lapor ke ponsel pengontrol
rjmp        kembali
lampu2_hidup: cbi       portb,1             ;pb1 bernilai 0 (clear),lampu 2 nyala
rcall       report
rjmp        kembali
lampu1_mati:  sbi       portb,0             ;pb0 bernilai 1 (set),lampu 1 mati
rcall       lapor               ;lapor ke ponsel pengontrol
rjmp        kembali
lapor:  rcall       report
            rcall       long_delay
rcall       report
ret
```

Gambar 8. Program untuk menentukan lampu mana yang dihidupkan atau dimatikan.

```
dial_nomor: .db "atd",0
stop_dial: .db      "at+chup",0x0d,0x0a, 0
report:   ldi       YLramadr,RamStart
dial:     ld        r0,Y+
          mov       txbyte,r0
          cpi       txbyte,""
          brne      dial
          ldi       zl,low(dial_nomor*2)
          ldi       zh,high(dial_nomor*2)
          rcall     send_command
dial1:    ld        r0,Y+
          mov       txbyte,r0
          cpi       txbyte,""
          brne      dial2
          ldi       txbyte,0x3b
          rcall     tulis_tx
          ldi       txbyte,0x0d
          rcall     tulis_tx
          ldi       txbyte,0x0a
          rcall     tulis_tx
          ser       temp
          rcall     delay1
          ser       temp
          rcall     delay1
          rcall     long_delay
          ldi       zl,low(stop_dial*2)
          ldi       zh,high(stop_dial*2)
          rcall     send_command
          ret
dial2:    rcall     tulis_tx
          rjmp      dial1
```

Gambar 9. Program prosedur *report.*

KESIMPULAN

Pengontrol lampu via *missed call* berbasis mikrokontroler AVR AT90S2313 yang dilaporkan pada makalah ini telah diuji dan dibuktikan dapat menghidupkan /mematikan enam buah lampu dari jarak jauh, dengan cara melakukan *missed call* dua kali. Banyaknya RING pada *missed call* pertama menspesifikasikan nomor lampu yang dipilih, sedangkan banyaknya RING pada *missed call* kedua menghidupkan (cacah RING=1) atau mematikan (cacah RING=2) lampu tersebut.

Missed call lebih murah dari pada SMS ataupun DTMF dan alat ini dapat dioperasikan dengan handphone GSM yang sudah sangat murah. Tetapi, jangkauan jaringan GSM merupakan salah satu keterbatasannya. Selain itu, alat belum mampu mendeteksi nomor HP pemanggil kecuali sesudah divonis sebagai *missed call.* Nomor HP *transceiver* harus dirahasiakan agar tidak sembarang orang dapat mengendalikan lampu tersebut. Banyaknya lampu yang dapat dikontrol dibatasi oleh cacah RING yang dibolehkan oleh operator.

DAFTAR PUSTAKA

Atmel. 2002. *8-Bit AVR Microcontroller with 2K Bytes of In-System Programmable Flash AT90S2313*. www.atmel.com [1 Oktober 2005].

Istiyanto JE & Efendy Y. 2004. Rancangan dan Implementasi Pengendali Jarak Jauh Berbasis Mikrokontroler AT89C52 dan Layanan SMS. *Jurnal Ilmu Dasar Vol 5*. **2**: 76-86.

Istiyanto JE & Purwadi E. 2005, *Alat Pemantau Suhu Jarak Jauh Berbasis SMS*. SN Universitas Ahmad Dahlan. Jogjakarta.

Istiyanto JE & Alrosyid. 2005. *A Prototype of a Device Control and Alarm Monitor System Based on the GSM Short Message Service and the AVR AT90S2313 Microcontroller*. TSSA 2005. Institut Teknologi Bandung.

Istiyanto JE & Hakim AR. 2007. A DTMF-based Remote Device Monitoring and Control, *Proceedings of The 2nd Jogja International Physics Conference* (JIPC). Universitas Gadjah Mada.

Istiyanto JE & Wibhowo FW. 2007. DC Motor Control Based on Cellular Phone. *Proceedings of The 2nd Jogja International Physics Conference (JIPC)*. Universitas Gadjah Mada.

Maxim. 2004. *+5V-Powered Multichannel RS-232 Drivers/Receivers*. Maxim.

Siemens. 2001. AT *Command Set Reference Manual*. Siemens.